Degree Final Project

Bachelor's degree in Industrial Technology Engineering

Design and implementation of a robotic application to facilitate the book gathering in a library

Author: Magí Dalmau Moreno
Director: Dr. Raúl Suárez Feijóo

Date: September 2019

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Abstract

This project contributes to solve the necessity of automating the gathering of the books in a library. In order to achieve this goal, a robotic solution is designed, implemented and tested. This solution is based on computer vision, strategic object-manipulation, autonomous navigation and intelligent motion-planning.

This project solves in particular the following task. The robot, placed in some point of a previously mapped room, gets a human order to get a book from the table in the room. Then, the robot navigates autonomously to a predefined position in front of the table and the robot detects the book. If the book is there and it is reachable by the robot from its current position, the robot calculates the proper path to pick the book off the table. Once the book has been picked, the robot returns to the initial position and offers the book to the human user, releasing it when detecting the user has grasped the book. It is important to highlight that the project focuses on the manipulation of the book and its localization through ArUco markers.

The developed algorithm that controls the robot is implemented in the class-oriented programming languages C++ and Python and it is mainly based on the Robot Operating System (ROS) framework. The robot TIAGo from PAL Robotics has been chosen to test the developed solution, both in simulated and real scenarios.

As a result of this project, a robot is able to calculate how it should alter the orientation and position of a book on a table in order to make the task of grasping with a parallel gripper possible. It is suggested to go further in this line by making the approach more robust to the presence of obstacles.

Contents

1. Introduction

1.1. Project objectives and motivation

In the last years, our society has been looking at robots as future helpers for humans: they could assist elderly people, cooperate with us in the execution of tedious and hard works, and even replace us for the accomplishment of the most dangerous tasks. This project intends to contribute in solving the necessity, in a library, of automating the gathering of the books. In order to achieve this goal, a robotic solution is designed and implemented. This solution is based on computer vision, strategic object-manipulation and intelligent motion-planning. The proposed generic solution is designed and it is validated through experiments with in a real robot of the Institute of Industrial and Control Engineering (IOC-UPC).

This project solves in particular the following task. The robot, placed in some point of a previously mapped room, gets a human order to get a book from the table in the room. Then, the robot navigates autonomously to a predefined position in front of the table. If the book is there and it is reachable by the robot from its current position, the robot detects the book and calculates the proper path to pick the book off the table. Once the book has been picked, the robot returns to the initial position and offers the book to the human user. In Figure 1.1, this task is outlined.

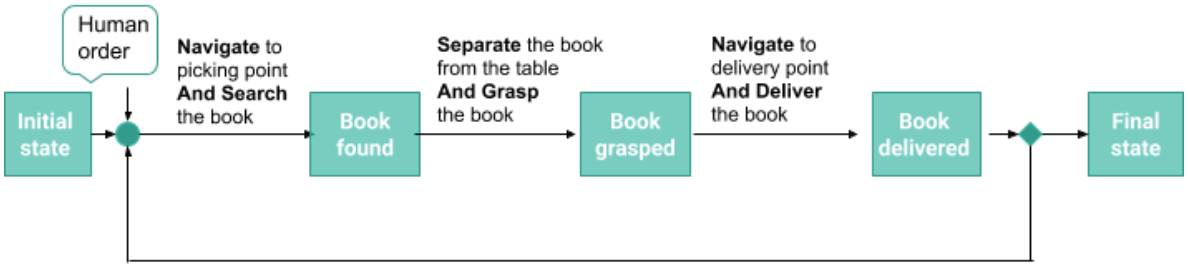

Figure 1.1 Task overview.

1.2. Project scope

The project is focused on a single task from all those which would be required in an automated library. Moreover, additional restrictions and requirements are set to adapt the dimension of the project and make it feasible in practice.

These restrictions and requirements are listed below:

- All the books have an identifying marker on both covers.
- There are no obstacles at the table where the book lies. Besides, the book does not protrude from the table. The distance between the margin of the table and the book is compatible with the arm of the robot.
- The robot has to be capable to grasp and manipulate any book of a common library.
- The books have a minimum thickness that makes it possible to push them on the table with the robot fingers but ensuring the fingers do not touch the table.
- The weight of the books does not exceed a given payload.
- The necessary information about the existing books and tables are available in a database.
- The room has been previously mapped for navigation.
- The robot will bring the book to the user at the initial location.

1.3. Resources

In order to solve the proposed task, different robotic solutions have been investigated, such as the use of AGVs or non-collaborative robots. However, the conclusion is that for a civic environment, such as a library, it is required a collaborative robot prepared to operate surrounded by people without danger and with a friendly appearance not to generate social rejection. Therefore, the developed solution is tested in a real TIAGo robot from the company PAL Robotics (shown in Figure 1.2). TIAGo is a service robot designed to work in indoor environments. TIAGo's features make it the ideal platform for research, especially on assisted living or light industry. It combines mobility, perception, manipulation and also human-robot interaction capabilities for one specific goal: to be able to assist in research. It is composed of a differential drive mobile platform that has a liftable torso with a 7-degrees-of-freedom arm mounted on top of it. The robot has a bi-articulated head with a RGB-D camera. In addition is provided of force torque sensor in the wrist, that detect collisions and interactions; a LIDAR to detect obstacles during the navigation; and a speaker that, through a text-to-speech service, allows the robot to talk.

The developed algorithm that controls the robot is implemented in the class-oriented programming languages C++ and Python and it is mainly based on the Robot Operating System (ROS) framework [1]. ROS is an open-source flexible framework that aims to simplify

the development of general-purpose collaborative software for robots. In addition, ROS encourages their users to contribute to the community with new software packages. Besides, it eases the collaboration between different research institutions by developing new software based on each other's work. Although ROS is not an operating system, it provides services designed for a heterogeneous computer cluster such as hardware abstraction, implementation of commonly used functionality, message-passing between processes, low-level device control, and package management. In the ROS framework, the processing takes place in nodes that are seamlessly distributed, in the same or in different cores and machines. These nodes, coordinated by a master node, send data streams to each other and can be dynamically configured. Although originally oriented to UNIX-like systems (e.g. Ubuntu), it is also adapted to work in other operating systems (e.g. macOS and Microsoft Windows). ROS has been the framework used to develop and to connect all the software modules designed for this work.

Figure 1.2 TIAGo Robot, PAL Robotics.

2. Solution development

In order to implement the functionalities proposed in the introduction, the developed algorithm follows the flow diagram schematized in the Figure 2.1. In the diagram, the system states are represented as blocks and the actions as arrows. Notice that all the loops have a limit of iterations and, if the limit is reached, the robot asks for user instructions.

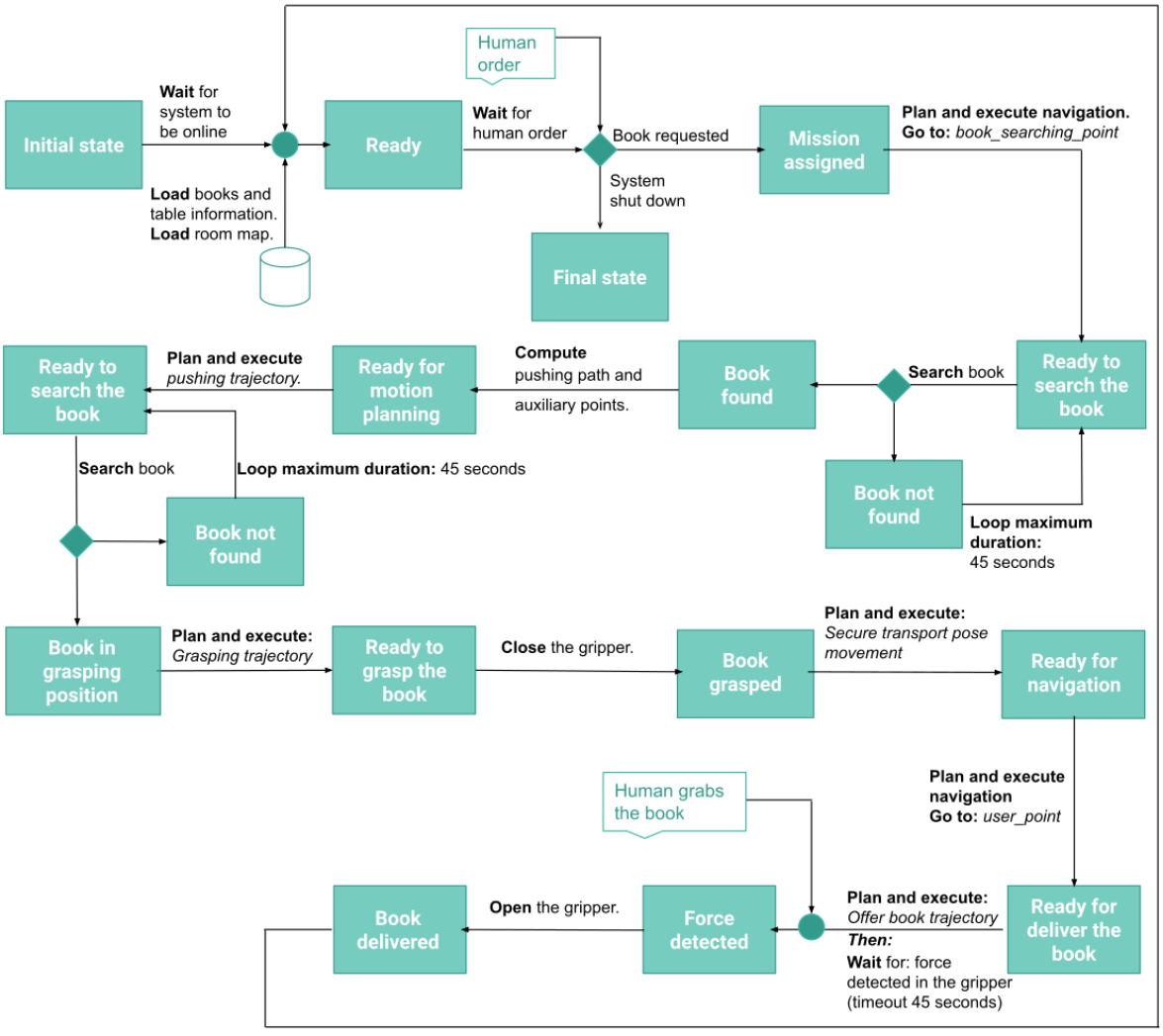

Figure 2.1 Solution overview.

Note that the proposed solution can be divided in independent subtasks each one involving perception, manipulation, user interaction or navigation. Then, for a better reader understanding, the explanation of the proposed approach has been divided into four sections, one for each involved robotics knowledge area.

2.1. Perception

To be able to pick up the book, the robot needs to know the pose of the book as well as the pose of the table where the book lies. In order to calculate these positions, ArUco markers [2], [3] (explained below) are placed on the books and the table. These markers are identified by the robot camera and their positions are treated through a set of transformations (TF) [4].

2.1.1. ArUco markers

An ArUco marker is a square-shaped marker that consists of a wide black border and an inner binary matrix which determines its identifier. The black border allows rapid detection in the image and the binary coding permits identification and the use of techniques to detect and correct errors.

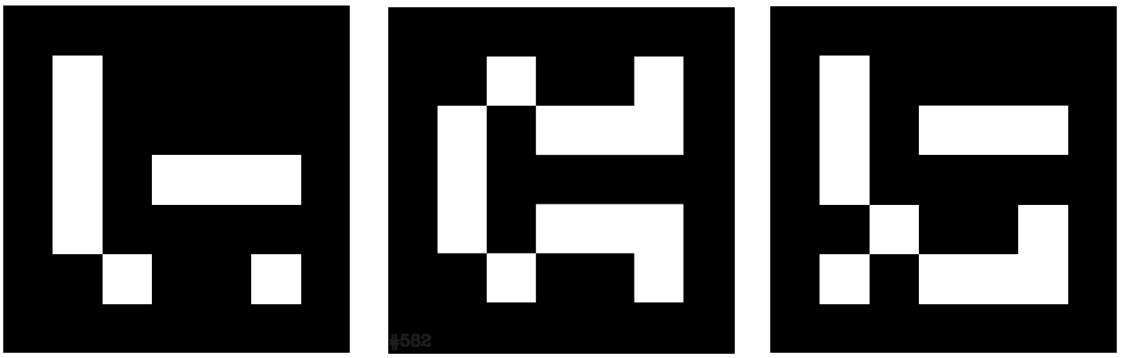

Figure 2.2 Example of ArUco markers.

The ArUco Library allows the computation of the position and orientation of a marker (of known dimensions) with respect to the robot camera as well as the marker numerical identification (id) that is used in this project for checking which object has been localized (each ArUco id is assigned to a single object). For this purpose, this library offers a node that connects to the image stream of the camera, and automatically identifies the markers that appears in the image. In addition, it calculates their positions and orientations with respect to the frame of the camera using the optical characteristics of the camera. With each new image, new marker information is obtained and published in a ROS topic so that other nodes can use it.

2.1.2. Developed perception model

Several TFs have been defined in the scenario in order to compute the book manipulation path. The most relevant TFs used to position the different elements of the scenario are represented in Figure 2.3.

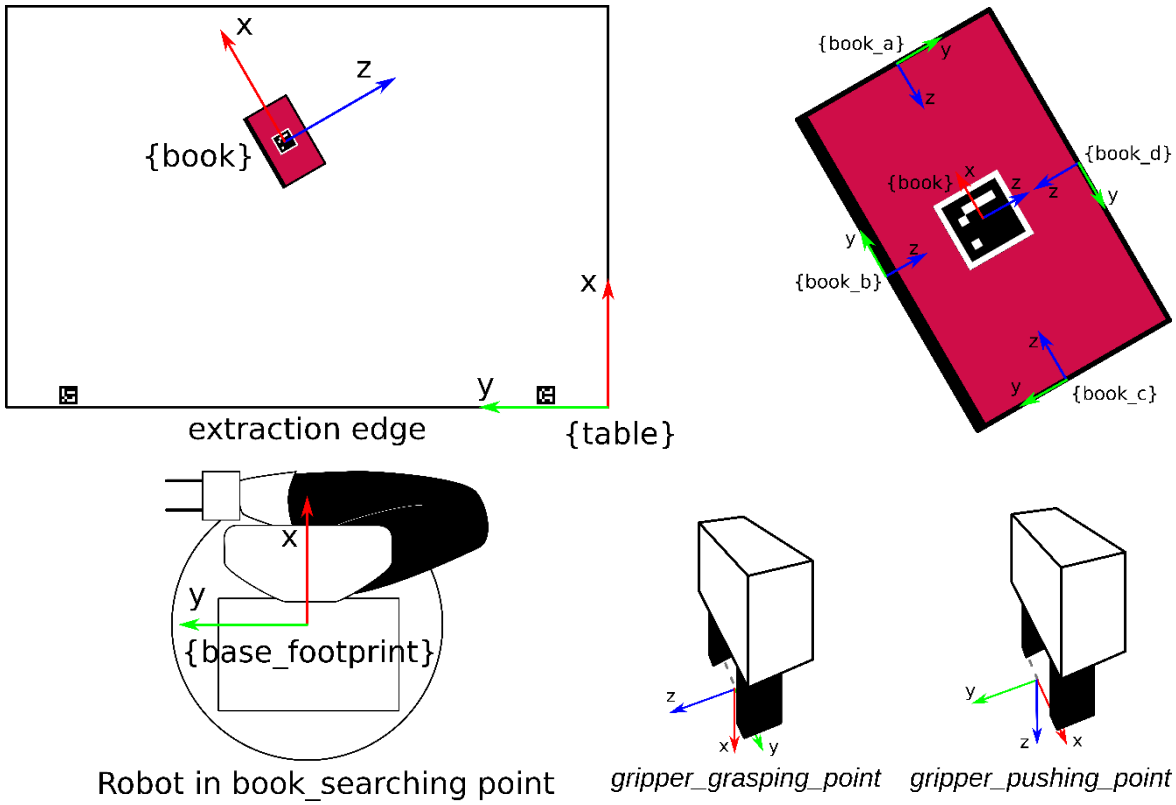

Figure 2.3 Reference systems TF used for the book location and manipulation

Note that the defined $\{base\_footprint\}$ and gripper frames are solidary to robot links and, then, directly known by forward kinematics [4]. The book frames $\{book\_a\}$ to $\{book\_d\}$ are fixed with respect to the {book} frame. Nonetheless, the positions of the table and the book are computed from the marker information as it is explained below.

2.1.2.1. Book transformation

As shown in Figure 2.3, each book has a (unique) marker from which, by means of the aforementioned procedure, the estimated position and orientation of the marker are obtained. This obtained pose is not exact so a refinement process is carried out in order to obtain a

low-error book pose. This process consists in two parts:

Correction of the transform rotation

The objective of this correction is to leave the perceived plane of the ArUco marker parallel to the plane of the floor (and therefore also parallel to the table and the book). To achieve this, a rotation $\mathrm{Rot}(\mathbf{n}, \theta)$ of an angle θ around an axis **n** is applied to the obtained raw TF.

Being $\boldsymbol{y}_{ar}^{ar} = [0, 1, 0]^T$ the y-axis of the ArUco expressed in its own base *(ar)* and being ${}^{bf}R_{ar}$ the raw rotation matrix of the ArUco TF with respect to the {base footprint} frame (bf), the raw y-axis of the ArUco in $\{base\_footprint\}$ is calculated as follows:

$$\boldsymbol{y}_{bf}^{ar} = {}^{bf}R_{ar}\, \boldsymbol{y}_{ar}^{ar} \qquad \text{(Eq. 2.1.1)}$$

By definition, in order for the plane of the ArUco and the plane of the ground to be parallel, their normal vectors must be also parallel and, therefore, their dot product must be equal to 1. Thus, being $\boldsymbol{y}_{bf}^{ar}$ and the z-axis of the robot ($\boldsymbol{z}_{bf}^{bf} = [0, 0, 1]^T$) perpendicular to the plane of the ArUco and of the ground, respectively, the angle between both should be 0. With this objective in mind, the angle between these two vectors is calculated by the equation:

$$\theta = arccos(\, \boldsymbol{y}_{bf}^{ar} \cdot \boldsymbol{z}_{bf}^{bf}) \qquad \text{(Eq. 2.1.2)}$$

The calculated θ angle is the angle that the raw TF must be rotated to achieve the objective. The rotation axis, **n**, must be perpendicular to the plane formed by both vectors $\boldsymbol{y}_{bf}^{ar}$ and $\boldsymbol{z}_{bf}^{bf}$. Therefore, the vector $\boldsymbol{n}$ is calculated as follows:

$$\boldsymbol{n} = \frac{\boldsymbol{y}_{bf}^{ar} \times \boldsymbol{z}_{bf}^{bf}}{\sin\theta} \qquad \text{(Eq. 2.1.3)}$$

Once the parameters $\boldsymbol{n}$ and θ have been obtained, the rotation $\mathrm{R}(\mathbf{n}, \theta)$ is applied to the raw TF to obtain the corrected book orientation:

$${}^{bf}R_{book} = \mathrm{Rot}(\mathbf{n}, \theta)\ {}^{bf}R_{ar} \qquad \text{(Eq. 2.1.4)}$$

Correction of the transform translation

Considering as negligible the thickness of the ArUco marker, the z-component (i.e. the vertical component) of the TF translation of the ArUco expressed in $\{base\_footprint\}$, ${}^{bf}\boldsymbol{p}_{ar}$, should correspond to the sum of the table height and the book thickness (both values known a priory). Hence, this value is directly assigned to the z-component of the book position ${}^{bf}\boldsymbol{p}_{book}$, as it is stated in the following equation:

$$ {}^{bf}\boldsymbol{p}_{book}|_z = \text{table\_height} + \text{book\_thickness} \qquad \text{(Eq. 2.1.5)} $$

Notice that the x- and y-components of ${}^{bf}\boldsymbol{p}_{book}$ remain equal to the x- and y-components of ${}^{bf}\boldsymbol{p}_{ar}$.

2.1.2.2. Table transformation

To calculate the TF of the table, there are two ArUcos positioned in the extraction edge of the table, as shown in Figure 2.3. This configuration has been established to reduce the error in the perception of the table orientation. If only an ArUco was used, a slight error in the ArUco orientation would produce a significant error in the table positioning, as the table is a big object (as opposed to the dimensions of a book). Below, it is explained how the transformation of the table is obtained from the two ArUcos.

Correction of the captured ArUcos transforms translation

First, a correction is applied to the translation components of the raw transforms of both ArUcos, ${}^{bf}\boldsymbol{p}_{ar\_L}$ for the left one and ${}^{bf}\boldsymbol{p}_{ar\_R}$ for the right one. This correction consists in positioning the markers at the table surface level (negligible marker thickness is assumed). For this purpose, the value of the table height is assigned to the z-component of each translation expressed in $\{base\_footprint\}$, as seen in the following equation:

$$ {}^{bf}\boldsymbol{p}_{ar\_L}|_z = {}^{bf}\boldsymbol{p}_{ar\_R}|_z = \text{table\_heigth} \qquad \text{(Eq. 2.1.6)} $$

Building the transform rotation

In order to construct the rotation matrix of the table transformation, note that the z-axis of

the table frame expressed in $\{base\_footprint\}$, $\boldsymbol{z}_{bf}^{table}$, is equal to the robot z-axis also expressed in $\{base\_footprint\}$, $\boldsymbol{z}_{bf}^{bf}$:

$$\boldsymbol{z}_{bf}^{table} = \boldsymbol{z}_{bf}^{bf} \quad \text{(Eq. 2.1.7)}$$

The y-axis of the table expressed in $\{base\_footprint\}$ is the vector that has as origin in the right ArUco and as end the left ArUco once normalized, as seen in the equation:

$$\boldsymbol{y}_{bf}^{table} = \frac{{}^{bf}\boldsymbol{p}_{ar\_L} - {}^{bf}\boldsymbol{p}_{ar\_R}}{\left\| {}^{bf}\boldsymbol{p}_{ar\_L} - {}^{bf}\boldsymbol{p}_{ar\_R} \right\|} \quad \text{(Eq. 2.1.8)}$$

Note that since the vertical component of the position of both ArUcos has been corrected in the previous step (i.e. both ArUcos have the same vertical position component), the mentioned vector $\boldsymbol{y}_{bf}^{table}$ is perpendicular to the vector $\boldsymbol{z}_{bf}^{table}$. Finally, the x-axis $\boldsymbol{x}_{bf}^{table}$ is computed as the cross product between both found axes as specified in the following equation:

$$\boldsymbol{x}_{bf}^{table} = \boldsymbol{y}_{bf}^{table} \times \boldsymbol{z}_{bf}^{table} \quad \text{(Eq. 2.1.9)}$$

Once calculated the three axes of the table frame expressed in $\{base\_footprint\}$, the computation of the rotation matrix is direct:

$${}^{bf}R_{table} = \begin{bmatrix} \boldsymbol{x}_{bf}^{table} & \boldsymbol{y}_{bf}^{table} & \boldsymbol{z}_{bf}^{table} \end{bmatrix} \quad \text{(Eq. 2.1.10)}$$

The origin of the table frame is defined at the table vertex closest to the right marker. The position of this marker is known with respect to the $\{base\_footprint\}$ frame (perceived by the camera) but also with respect to the $\{table\}$ frame (a setting of the environment). Then, it is hold that

$$\begin{bmatrix} {}^{bf}\boldsymbol{p}_{ar\_R} \\ 1 \end{bmatrix} = {}^{bf}T_{table} \begin{bmatrix} {}^{table}\boldsymbol{p}_{ar\_R} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{bf}R_{table} & {}^{bf}\boldsymbol{p}_{table} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} {}^{table}\boldsymbol{p}_{ar\_R} \\ 1 \end{bmatrix} \quad \text{(Eq. 2.1.11)}$$

Developing the last equation, ${}^{bf}\boldsymbol{p}_{table}$ is obtained as:

$${}^{bf}\boldsymbol{p}_{table} = {}^{bf}\boldsymbol{p}_{ar\_R} - {}^{bf}R_{table}\,{}^{table}\boldsymbol{p}_{ar\_R} \quad \text{(Eq. 2.1.12)}$$

Hence, both translation and orientation components of the table transformation have been computed.

2.2. Manipulation

With the robot in front of the table and with all the positions and orientations described in section 2.1 obtained, the book must be moved to a position and orientation where it can be picked up. The book lies in a random position on the table, as it is shown in Figure 2.4. The book is at a certain angle and distance with respect to the $extraction\_edge$ of the table, which is the table edge in front of the robot. The aim in this point is to place the book parallel to the extraction edge and that an experimentally-determined $cantilever\_distance$ of the book comes out of the table edge (so that it can be picked up but without falling down first). Once the gripper is positioned in the $pre\_pushing$ position, it performs a circular trajectory in order to place the book parallel to the $extraction\_edge$. Then, a straight path is followed to bring the book closer to the edge of the table without changing the book orientation. Afterwards, the gripper is placed at the $pre\_grasping$ position and it attempts the book grasping.

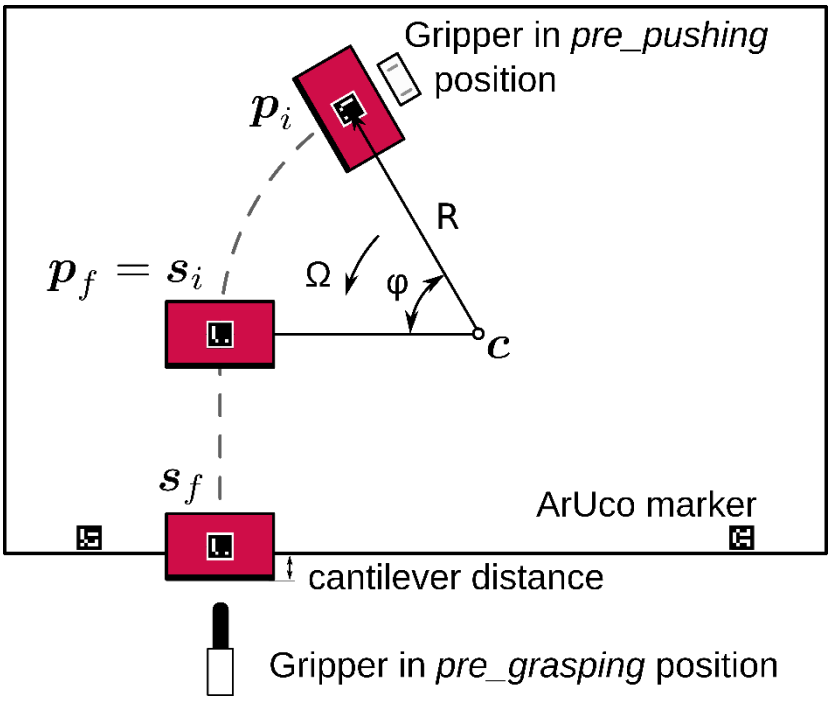

Figure 2.4 Pushing trajectory of the book.

2.2.1. Calculation of the circular pushing path

2.2.1.1. Initial pushing point

Each book registered in the system has associated points $book\_a, book\_b, book\_c, book\_d$

relative to the ArUco marker placed on the book (for more information, see section 2.1.2). The book is pushed from the point farthest from the extraction edge (i.e. the one with the largest x-component expressed in the coordinates of the table), since it minimizes the angle to be rotated. In addition, only the two long sides are considered to facilitate the book rotation, i.e. only point $book\_b$ and $book\_d$ are taken into account. This initial pushing point is henceforth called $\boldsymbol{p_i}$ and its associated transform is called ${}^{table}T_{\boldsymbol{p_i}}$.

The chosen initial point $\boldsymbol{p_i}$ for each book orientation can be seen in Figure 2.5. Note that in the extreme case where both points are at the same distance (i.e. the book has one of its long sides perpendicular to the extraction edge), the point closest to a lateral table edge is chosen, as it allows more space to perform the pushing path.

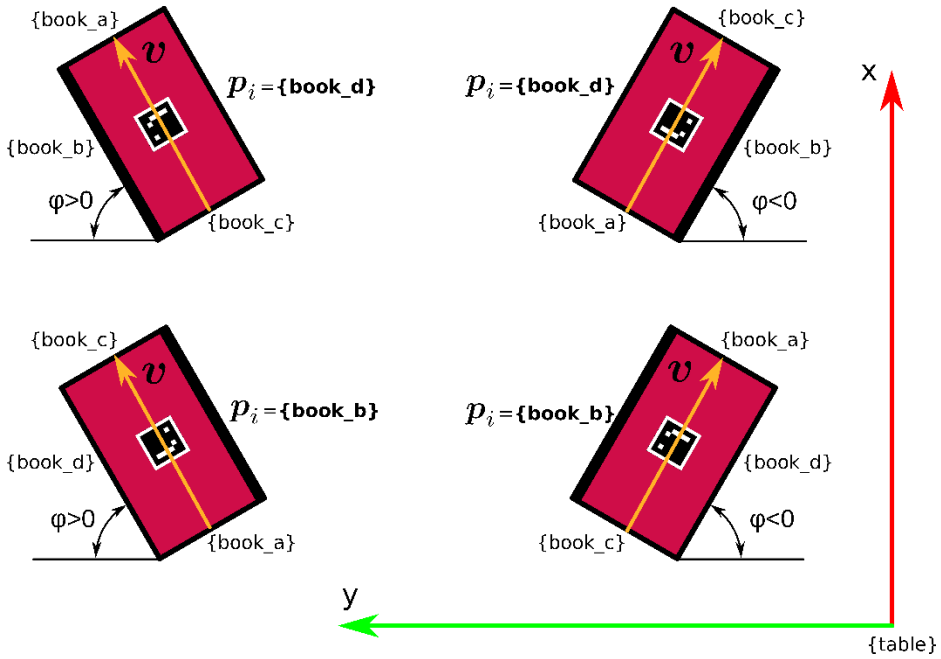

Figure 2.5 Possible book orientations and pushing point selection.

2.2.1.2. Rotation angle

The angle φ that the book should be rotated is the one described between the book mold and the extraction edge. This angle is positive if the book must be rotated anticlockwise and negative otherwise. Due to the rectangular shape of the book, the line described between points $book\_a$ and $book\_c$ is the same angle with the extraction edge. As the position of

points $book\_a$ and $book\_c$ can be easily calculated, they are used to determine the angle of rotation.

A vector $\boldsymbol{v}$ is calculated according to the position of the book (see Figure 2.5). The vector origin is the point $book\_a$ or $book\_c$ that has the smallest x-component expressed in coordinates of the table (i.e. the one closest to the extraction edge). The vector end is the other point $book\_a$ or $book\_c$, i.e. the furthest one from the extraction edge. Note that the extreme case in which $book\_a$ and $book\_c$ are at the same distance the angle to rotate is zero because the longest sides of the book are already parallel to the extraction edge. Once the vector $\boldsymbol{v}$ has been calculated (note that it is expressed in table coordinates), the rotation angle is computed as:

$$\varphi = \arctan \frac{\boldsymbol{v}|_x}{\boldsymbol{v}|_y} \qquad \text{(Eq. 2.2.1)}$$

Note that according to the orientation of the book the angle lies within the following range:

$$-\frac{\pi}{2} \leq \varphi \leq \frac{\pi}{2} \qquad \text{(Eq. 2.2.2)}$$

2.2.1.3. Circular path radius and center.

The radius of the circular path is computed taking into account the following requirements (see Figure 2.6 for a better understanding):

1. The gripper has to maintain the contact with the book at the two fingers all the time. Then, the velocities $\boldsymbol{v}_L$ and $\boldsymbol{v}_R$ of the left and right fingers, respectively, expressed in the $\{table\}$ reference must have both the x-component negative.
2. To make the design more robust and, hence, ensure the contact at both fingers, the velocities $\boldsymbol{v}_L$ and $\boldsymbol{v}_R$ are not allowed to differ more than a given a relation K_{vel} which is experimentally determined.
3. The upper bound of the radius R of the extraction trajectory is the maximum that allows the complete trajectory execution without the book falling out of the table.
4. The trajectory must minimize the movement in the y-axis with respect to the table reference. This condition increases the chances of success even if the book is close to one of the lateral edges of the table (and for that reason with a risk of falling when executing the circular trajectory).

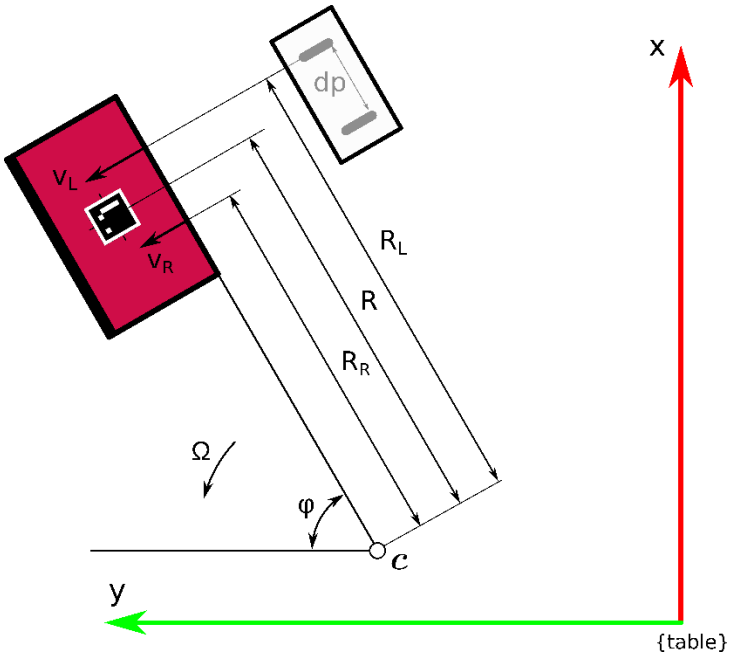

Figure 2.6 Circular path radius and center determination.

In order to fulfill these requirements, taking into account that the gripper point placed between the fingers follows a circle of radius R with an angular velocity Ω, the turning radius of the left and right fingers, R_L and R_R respectively, as well as the velocities $\boldsymbol{v}_L$ and $\boldsymbol{v}_R$ (view Figure 2.6) are computed as follows

$$R_L = R + \frac{dp}{2} \qquad \text{(Eq. 2.2.3)}$$

$$R_R = R - \frac{dp}{2} \qquad \text{(Eq. 2.2.4)}$$

$$\boldsymbol{v}_L = \Omega\, R_L \qquad \text{(Eq. 2.2.5)}$$

$$\boldsymbol{v}_R = \Omega\, R_R \qquad \text{(Eq. 2.2.6)}$$

where dp is the distance between the fingers when fully open.

To meet condition 2, it must be hold that:

$$\boldsymbol{v}_L \leq K_{vel}\, \boldsymbol{v}_R \qquad \text{(Eq. 2.2.7)}$$

Then solving for R, it is obtained:

$$R \geq \frac{\frac{dp}{2}(1+K_{vel})}{(K_{vel}-1)} \quad \text{with } K_{vel} > 1 \qquad \text{(Eq. 2.2.8)}$$

To satisfy the 3rd condition, the end point of the path, $\boldsymbol{p}_f$ (see Figure 2.4), must accomplish that:

A. In order to prevent the book from falling down the extraction edge, at most the desired cantilever percentage of the book protrudes.

$$\boldsymbol{p}_f|_x \geq (1-cantilever)\ book\_length \qquad \text{(Eq. 2.2.9)}$$

B. When the angle φ is positive, the book is pushed to the left table edge. To prevent the book from falling off the left edge, the following inequality must be hold:

$$\boldsymbol{p}_f|_y \leq table\_width \qquad \text{(Eq. 2.2.10)}$$

C. When the angle φ is negative, the book is pushed to the right table edge. To prevent the book from falling off the right edge, the following inequality must be hold:

$$\boldsymbol{p}_f|_y \geq 0 \qquad \text{(Eq. 2.2.11)}$$

The end point $\boldsymbol{p}_f$ can be expressed as a function of the starting point $\boldsymbol{p}_i$, the radius R and the rotation angle φ as:

$$\boldsymbol{p}_f|_x = \boldsymbol{p}_i|_x - R\sin(|\varphi|) \qquad \text{(Eq. 2.2.12)}$$

$$\boldsymbol{p}_f|_y = \boldsymbol{p}_i|_y + R\ \text{sign}(\varphi)\ (1-\cos(\varphi)) \qquad \text{(Eq. 2.2.13)}$$

By solving the inequalities (that assure the fulfillment of the 3rd condition) with the expressions of $\boldsymbol{p}_f$ found, the following restrictions are obtained for the radius R:

$$R \leq \frac{\boldsymbol{p}_i|_x - (1-cantilever)\ book\_length}{\sin(|\varphi|)} \qquad \text{(Eq. 2.2.14)}$$

$$R \leq \begin{cases} \dfrac{table\_width \; - \; \boldsymbol{p}_i|_y}{1 - \cos(\varphi)} & \textbf{if } \; \varphi > 0 \\ \dfrac{\boldsymbol{p}_i|_y}{1 - \cos(\varphi)} & \textbf{if } \; \varphi < 0 \end{cases} \qquad \text{(Eq. 2.2.15)}$$

Note that it is possible that the system of inequalities that govern the limits of the radius has no solution because the initial position of the book is at a point too close to an edge of the table, in that case it would not be possible to execute the trajectory and therefore take the book. This case is outside the assumptions that rule the project's scope and it is proposed to design, in subsequent works, a path prior to the extraction to relocate the book so that the conditions can be met to find a viable extraction path. If there is an interval of values that can take the radius respecting the calculated limits, to satisfy the fourth condition, the one with the minimum value is taken.

Once the radius R , the starting point $\boldsymbol{p}_i$ and the rotation angle φ have been calculated, as can be deduced in Figure 2.4 by trigonometry, the center $\boldsymbol{c}$ of the circular path is:

$$\boldsymbol{c} = \boldsymbol{p}_i - \text{sign}(\varphi)\, R \begin{bmatrix} \sin(\varphi) \\ \cos(\varphi) \end{bmatrix} \qquad \text{(Eq. 2.2.16)}$$

As mentioned above, the rotation angle φ is in the interval $-\frac{\pi}{2} \leq \varphi \leq \frac{\pi}{2}$.

2.2.1.4. Circular path points

The path that the $gripper\_pushing\_point$ (see Figure 2.3) must follow to place the book at the target position and orientation is composed of a sequence of $n + 1$ transformations. The k-th pose ${}^{table}T_{\boldsymbol{p}_k}$ is calculated as a rotation of angle φ_k and center $\boldsymbol{c}$ of the starting pose ${}^{table}T_{\boldsymbol{p}_i}$:

$${}^{table}T_{\boldsymbol{p}_k} = \begin{bmatrix} \text{Rot}(z, \varphi_k) & (\text{I} - \text{Rot}(z, \varphi_k))\, \boldsymbol{c} \\ \boldsymbol{0} & 1 \end{bmatrix} {}^{table}T_{\boldsymbol{p}_i} \;\; \text{with } \varphi_k = k\, \frac{\varphi}{n} \qquad \text{(Eq. 2.2.17)}$$

being I the identity 3x3-matrix.

So the entire circular path is:

$$\mathcal{P}_{circular} = \left\{ {}^{table}T_{\boldsymbol{p}_k} \mid k = 0 \dots n \right\} \quad \text{(Eq. 2.2.18)}$$

Note that $\varphi_{k=0} = 0$ and $\varphi_{k=n} = \varphi$.The number of steps n is calculated as follows

$$n = \left\lceil \frac{\varphi}{\Delta\varphi} \right\rceil \quad \text{(Eq. 2.2.19)}$$

and it depends on the discretization angle $\Delta\varphi$ which is experimentally determined.

2.2.2. Calculation of the rectilinear pushing path

2.2.2.1. Initial point

The initial point of the rectilinear pushing path $\boldsymbol{s}_i$ is the last point $\boldsymbol{p}_f$ of the circular trajectory (calculated in section 2.2.1 and expressed in the $\{table\}$ reference). Then, the associated poses of both points are also the same ($^{table}T_{\boldsymbol{s}_i} = {}^{table}T_{\boldsymbol{p}_f}$). Note that if the book is perfectly aligned with the table at the start of the pushing operation, the circular path is not needed and, in consequence, not computed. In this case, $\boldsymbol{p}_f$ is equal to $\boldsymbol{p}_i$ (and also the associated poses are equal).

2.2.2.2. Distance to cover

Being $\boldsymbol{s}_i|_x$ the x-component of the initial point of the rectilinear path in the $\{table\}$ reference, the distance to cover by the gripper to push the book to the target position (see Figure 2.4) is:

$$d_s = \boldsymbol{s}_i|_x - (1 - cantilever)\, book_{length} \quad \text{(Eq. 2.2.20)}$$

2.2.2.3. Rectilinear path points

The path that the $gripper\_pushing\_point$ (see Figure 2.3) must follow to place the book at the target pose is composed of a sequence of $m + 1$ transformations. The k-th pose of this sequence, $^{table}T_{\boldsymbol{s}_k}$, is calculated translating the starting pose $^{table}T_{\boldsymbol{s}_i}$ a distance d_k as

follows:

$$ {}^{table}T_{s_k} = \begin{bmatrix} \mathrm{I} & [-d_k, 0, 0]^T \\ \mathbf{0} & 1 \end{bmatrix} {}^{table}T_{s_i} \quad \text{with } d_k = k\,\frac{d_s}{m} \qquad \text{(Eq. 2.2.21)} $$

being I the identity 3x3-matrix.

So the entire rectilinear path is:

$$ \mathcal{P}_{straight} = \left\{ {}^{table}T_{s_k} \mid k = 0 \dots m \right\} \qquad \text{(Eq. 2.2.22)} $$

Note that $\mathrm{d}_{k=0}$ = 0 and $\mathrm{d}_{k=n}$ =d_s. The number of steps m is calculated as follows

$$ m = \left\lceil \frac{d_s}{\Delta \mathrm{s}} \right\rceil \qquad \text{(Eq. 2.2.23)} $$

and it depends on the discretization step Δs which is experimentally determined.

2.2.3. Calculation of the grasping point

When the grasping point is being computed, the working scenario is as follows: the robot has moved back a distance of 30 centimeters from the position it had during the book positioning operations. This movement took place in order to ensure the maneuverability necessary to perform the grasping movements. In addition, as neither the execution of the book positioning nor the movement of the robot base are perfect, the process of book and table localization described in section 2.1 is executed again.

The grasping point is the pose at which the $gripper\_grasping\_point$ must be positioned (see Figure 2.3) so that if the gripper closes the fingers, the book is safely grasp. Since the pushing operation leaves the book with its long sides parallel to the table extraction edge, the book grasp is attempted from the book side opposed to the initial pushing point (see section 2.2.1.1). The book point on this side, is called from now on as reference point and its pose with respect the book frame is called ${}^{book}T_{reference\_point}$ (i.e. the reference point is $\{book\_b\}$ if the book is pushed from $\{book\_d\}$ and vice versa). Then, the pose of the grasping point is calculated by applying a translation to the reference point as follows.

The reference point is displaced a distance d_{pen} towards the book, being computed as:

$$d_{pen} = penetration\_ratio \cdot cantilever \cdot book\_length \quad \text{(Eq. 2.2.24)}$$

being the $penetration\_ratio$ an experimentally determined parameter. In addition, the reference point is displaced upwards a distance d_{cent} with the objective of centering the space between the gripper fingers with the book. This distance is computed as

$$d_{cent} = \frac{book\_thickness}{2} + {}^{book}\boldsymbol{p}_{reference\_point}|_y \quad \text{(Eq. 2.2.25)}$$

being ${}^{book}\boldsymbol{p}_{reference\_point}|_y$ the y-component of the position of the $reference\_point$ pose expressed in the book frame. Consequently, the grasping point pose ${}^{book}T_{grasping\_point}$ is calculated as:

$${}^{book}T_{grasping\_point} = {}^{book}T_{reference\_point}\,{}^{reference\_point}T_{grasping\_point} \quad \text{(Eq. 2.2.26)}$$

with:

$${}^{reference\_point}T_{grasping\_point} = \begin{bmatrix} \mathrm{I} & [d_{cent}, 0, d_{pen}]^T \\ \mathbf{0} & 1 \end{bmatrix} \quad \text{(Eq. 2.2.27)}$$

2.2.4. Calculation of the auxiliary points

During the picking up process there is some uncertainty. This incertitude can be a source of malfunctions. In particular, the possible variations in the recognition of the position of the book and the table, the possible deviations in the behavior of the book during its pushing and the precision in the execution of the robot movements have been considered. All these factors imply that it is not possible to fully guarantee that the initial and final positions of each trajectory are exact. For example, a deviation in the detection of the position of the book could lead to an erroneous initial pushing point inside the book. This fact would lead to a collision with the book if the gripper went directly to that point. Another example could be that the book had not followed exactly the calculated trajectory during the pushing. In that case, if the gripper were separated from the book directly, an unwanted contact could occur.

Consequently, in order to make the picking up strategy of the book more robust, a set of auxiliary points has been defined which make the approaches to the initial trajectory points and the separation of the final trajectory points smoother and safer. The defined auxiliary points are listed below.

Two points $pre\_pushing\_point$ and $pre\_pushing\_point\_up$ have been added at the beginning of the circular pushing path. The $pre\_pushing\_point$ is computed moving back 5 cm the $initial\_pushing\_point$ and the $pre\_pushing\_point\_up$ is computed moving up 10 cm the $pre\_pushing\_point$.

After the pushing path, two points $post\_pushing\_point$ and $post\_pushing\_point\_up$ have been added. The $post\_pushing\_point$ is computed moving back 5 cm the $final\_pushing\_point$ and the $post\_pushing\_point\_up$ is computed moving up 10 cm the $post\_pushing\_point$.

In addition, a point $pre\_grasping\_point$ have been added before the grasping point. The $pre\_pushing\_point$ is computed moving back 4 cm the $grasping\_point$.

2.2.5. Trajectory planning and execution.

Within the book manipulation, two types of robot movements are identified: movements free in the joint configuration space and movements constrained in the Cartesian space. Both type of robot motions are planned using the MoveIt motion planning framework [5].

In order to plan the first type of movements, the RRT-Connect planning algorithm is used [6]. This sampling-based planner has been chosen because of its high versatility and the demonstrated good performance in complicated environments and highly-articulated robots, as is the case. The basic idea of the RRT-Connect planner is to build two sample trees, rooted at the initial and final joint configurations respectively. The motion planner grows the trees towards each other by iteratively adding to the sample trees new collision-free joint configurations. Then, while exploring the configuration space, the sample trees finally connect and a collision-free path connecting the start and final configurations is obtained.

In order to plan the second type of movements, the Descartes motion planning algorithm is used [7]. This planner receives as input the initial joint configuration and the Cartesian path that the end-effector must follow. The main idea of the algorithm is to iteratively obtain new inverse-kinematics solutions [4] for each point of the provided Cartesian path and try to connect them to the solutions found for the previous and next path points. A path solution is found when a collision-free path passing through a joint configuration associated to each path point exists, starting from the provided initial joint configuration.

2.2.5.1. Collision avoidance

In order to avoid collisions with the scenario elements, restricted areas are defined for the robot. Note that the robot restrictions has been already implemented in order to avoid self-collisions. As it is explained above, when a motion planner is launched, it tries to compute a trajectory that brings the robot from the initial to the final poses avoiding the restricted areas. In the present application, these zones are modeled as a box. The size of this box is different for each phase of the picking task. For the first phase, in which the robot must move to the initial point of the pushing trajectory, the box covers the table plus an extra height equal to twice the book thickness, so as not to collide with the table nor with the book, which lies within the table footprint. During the pushing trajectory, the box only covers the table, since the interaction with the book is obviously necessary. Afterwards, when the robot moves to the initial point of the grasping trajectory, the box covers the table plus an extra frontal space equal to twice the book protruding length and an extra height equal to twice the book thickness in order to avoid the collisions with the table and the book, which in this phase protrudes from the front of the table.

2.2.5.2. Book positioning phase

The planning and execution of the book positioning movements have been divided into three parts: *approach*, *push* and *retreat*.

The *approach* part is a free movement in the joint configuration space from the initial pose of the robot to the $pre\_pushing\_point\_up$. The *push* part is movement constrained in the Cartesian space. It starts in the $pre\_pushing\_point\_up$ and contains the rectilinear movement to the $pre\_pushing\_point$, the circular and the rectilinear pushing paths and it ends in the $post\_pushing\_point$. Finally, the *retreat* part is a free movement in the joint configuration space that starts in the $post\_pushing\_point$ and ends in the $post\_pushing\_point\_up$.

It should be noted that a Cartesian movement is more restricted than a free movement. For example, it could be the case that from the same initial pose in the Cartesian space, only a reduced subset of the total number of joint configurations associated to that initial pose allow the complete execution of the Cartesian movement. Hence, it is important to find a proper initial joint configuration for the *push* path described above. Consequently, a scheme has been designed to allow the selection of those joint configurations that make the *approach*, *push* and *retreat* movements compatible.

The planning scheme goes as follows:

- Step 1: A random joint configuration is generated.
- Step 2: The inverse kinematics is solved to place the gripper at the $pre\_pushing\_point\_up$ pose using the configuration generated in Step 1 as seed value. The result of this calculation is a joint configuration. If no solution is found, the process jumps back to Step 1.
- Step 3: The robot motions to reach the configuration found in Step 2 are planned in the joint configuration space avoiding collisions. If no plan is found, the process returns to Step 1.
- Step 4: The joint configuration calculated in Step 2 is used as the initial joint state and a joint trajectory is planned to follow the Cartesian pushing movement. The planner returns a plan along the Cartesian path avoiding collisions. If no plan is found, the process returns to Step 1.
- Step 5: The retreat movement is planned starting of the last joint state of the trajectory plane in the previous step. If no path avoiding collisions is found, the process returns to Step 1.

Note that the proposed planning scheme ends with *approach*, *push* and *retreat* movements that, once concatenated, provide a continuous and smooth pushing trajectory. In addition, notice that since random joint configurations are used as seed for the inverse-kinematics (IK) solving, the planning proceed avoids getting stuck in local minima.

Regarding the trajectory execution, it is important to highlight that velocity and acceleration restrictions are set with the objective of improving the execution of the pushing movement and avoid jerky movements within the book interaction.

2.2.5.3. Book grasping phase

The planning and execution of the book grasping movements have been divided into two parts: *approach* and *grasping*. The *approach* part is a free movement from the $post\_pushing\_point\_up$ to the $pre\_grasping\_point$. The *grasping* part is a Cartesian movement from the $pre\_grasping\_point$ to the $grasping\_point$.

In this phase, the Cartesian movement results in a rectilinear path with a length of 4 cm. Thus, it is simple enough and do not to require a planning scheme as described above. Considering

this in order to plan the movements, the following process has been followed:

- First, the *approach* motion is planned avoiding collisions (until a solution is found) and directly executed.
- Then, the *grasping* part is planned avoiding collisions (until a solution is found) and directly executed.
- Once the $grasping\_point$ is reached, the robot closes the gripper fingers, completing the grasping phase.

2.3. User interaction

The robot must help people to gather books in a library. Accordingly, the robot has to interact with the person it is helping. This interaction is framed in three contexts: Requesting a book to the robot, supply of the book collected by the robot to the person and notification of the progress or any problems to the user.

In order to request the book, an interface has been implemented. The communication with the system is done with a keyboard and is able to load those books that have stored as available in the library as well as the library tables. When a book is selected, the robot performs all the necessary processes to obtain it. Once the robot has picked up the book, it must deliver it to the user. To do this, the following scheme is executed once the robot is in front of the user, assuming that user has not changed its location. The arm-torso of the robot executes a pre-planned movement from the $safe\_transport\_pose$ (explained in section 2.4) to the pose for offering the book. It waits (with a timeout) until the wrist force-torque sensor detects that a force is being exerted (i.e. the user is trying to grasp the book). When the force exceeds a given threshold (experimentally determined), the robot opens the gripper.

During the whole process, the robot communicates its status in key steps, such as when it is ready to go to look for the requested book or when it offers the book to the user. In addition, whenever a problem arises, the robot asks for help. For this purpose, the voice interface of the TiaGO robot is used [8], to emit the messages prepared for each situation.

2.4. Navigation

To move between the point where the user is located and the table where the book lies, the robot needs to perform navigation tasks. For this purpose, the functionalities already implemented in the TIAGo robot are used [8]. The navigation software supplied with TIAGo can be viewed as a black box with the desired goal pose as input and the goal achievement status as output. The system developed in this project communicates with the navigation software through a ROS topic. As stated in the project assumptions (see section 1.2), the system is provided with an environment map where the robot operates and when the robot starts it is already located in the map.

When a book is requested, the location of the table where the book lies is given to the navigation service as input. Then, the robot executes its planning routines taking into account its initial position, the target and the static obstacles that appear in the map. When a path to the target position is found, the navigation begins. During all navigation the laser sensors and the RGB-D camera provide information of the environment to the navigation software. This dynamic feedback allows to update the obstacles within the planned path and modify it consequently. If no path is found, the robot asks for help from nearby people as explained in section 2.3.

To improve safety, the system moves the arm-torso of the robot into a specific position before executing any navigation (i.e. both navigating from the user position to the table and back). This position, called $safe\_transport\_pose$, keeps the torso fully lowered and the arm close to the body to lower the center of gravity and reduce the robot footprint. Therefore, the probability of impacts is reduced and the robot stability is improved. This position is compatible with the book transportation.

3. Results

The developed solution has been validated first in a simulation environment and then in a real scenario to avoid risks in the laboratory with the real robot. The implemented code and the videos from the experimentation can be found at gitioc.upc.edu/magi.dalmau/tiago_librarian.

3.1. Simulation results

The main part of the project has been simulated (i.e. the detection of the book, its positioning and its grasping). The used simulator is Gazebo [9] and it is provided with the model of the robot supplied by PAL Robotics. The simulated environment consists in a table with two ArUcos placed to the right and left and a book with an ArUco in its center. Several book and table positions and orientations have been tested in order to test the algorithm robustness. Figure 3.1, Figure 3.2, Figure 3.3, Figure 3.4 and Figure 3.5 shows snapshots of a simulated example of these experiments. It is concluded that the robot had been able to identify the position of the book and the table, generate and execute the pushing path (which places the book on the edge of the table), find the book again, grasp the book and move to the safe transport position. The simulation results have been a success and it has therefore been possible to carry out the experimentation on the real robot.

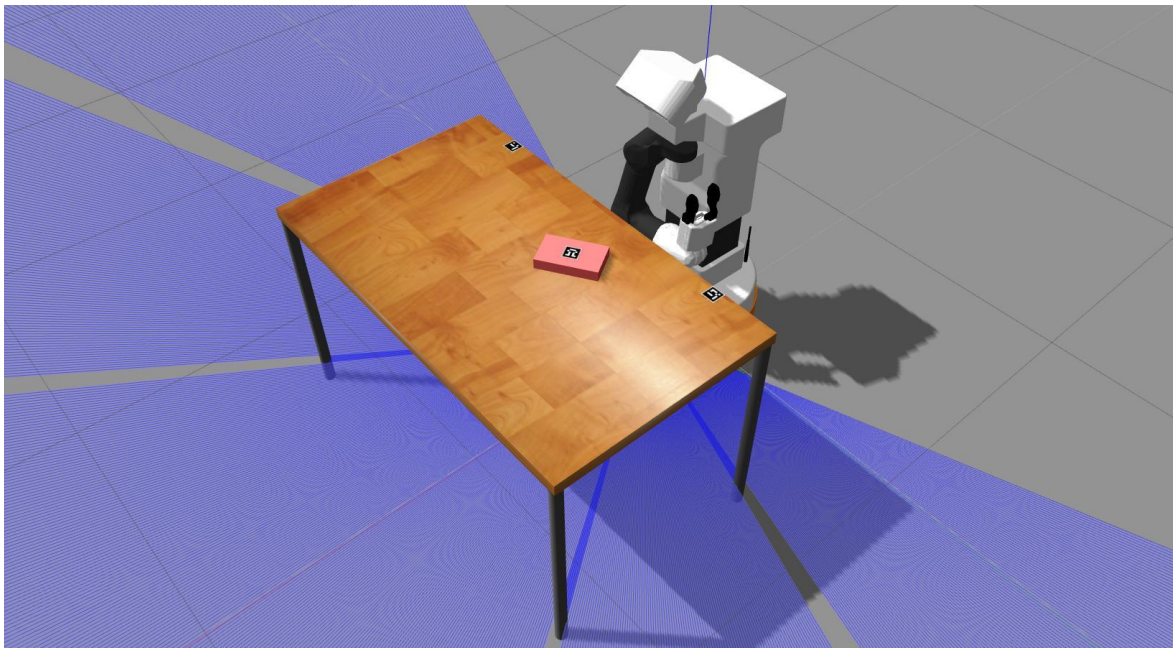

Figure 3.1 Robot searching the book.

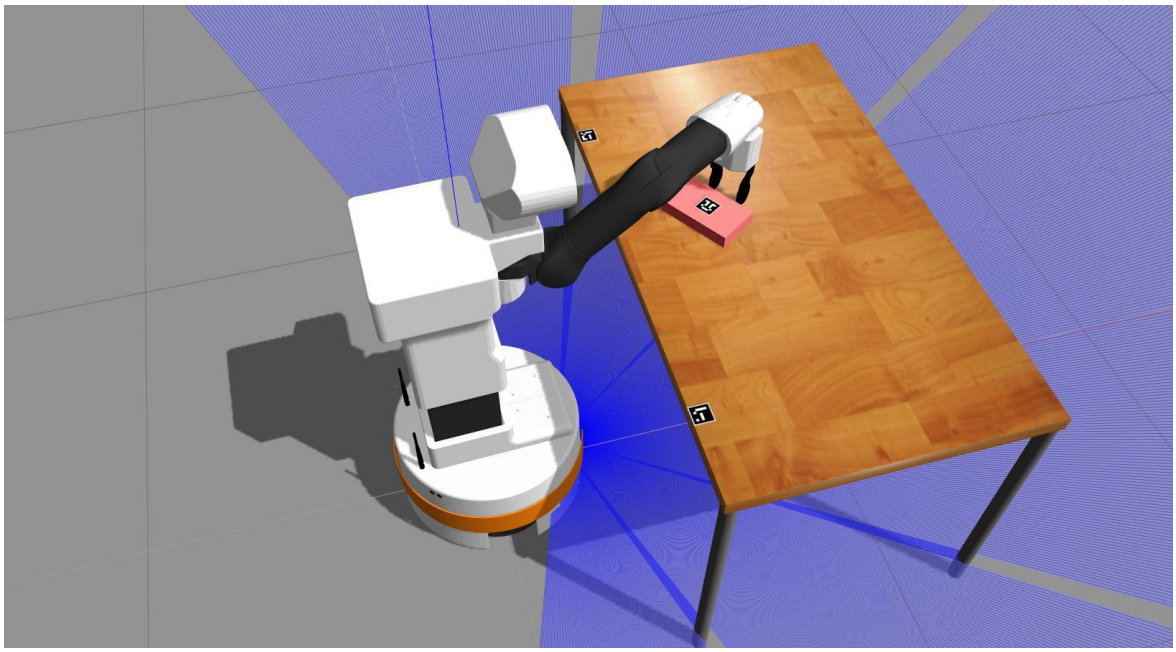

Figure 3.2 Robot performing the pushing trajectory.

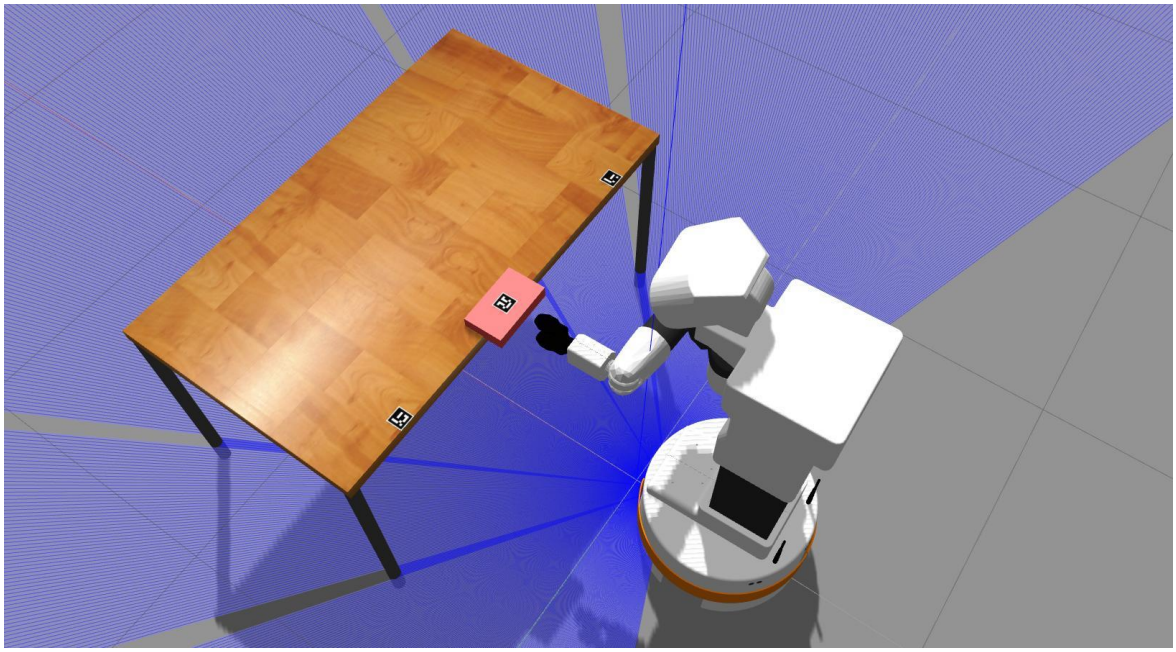

Figure 3.3 Robot ready for grasping the book.

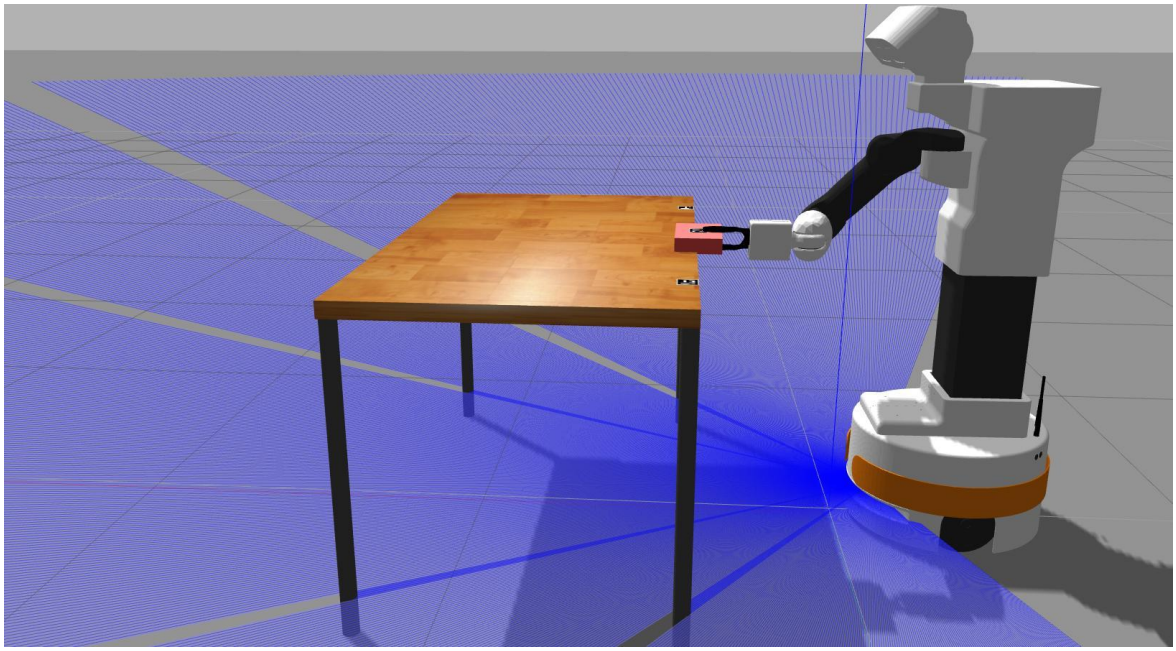

Figure 3.4 Book grasped.

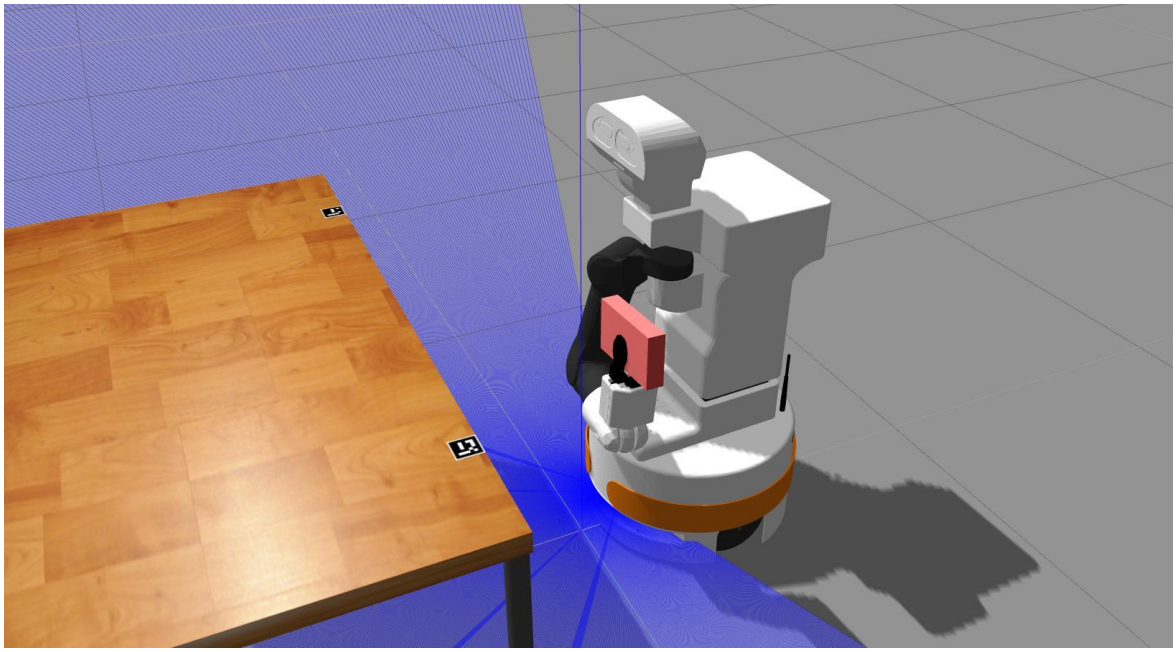

Figure 3.5 Robot in safe transport pose.

3.2. Experimentation with the real robot results

After the promising results obtained in the simulations, the complete proposed approach has been tested in the laboratory with the real robot. During the experimentation, multiple orientations, positions and book types have been tested in order to verify that it is a robust solution. Figure 3.6, Figure 3.7, Figure 3.8, Figure 3.9, Figure 3.10, Figure 3.11 and Figure 3.12 show pictures of a real example of these experiments. The results of the real experimentation agree with the results of the simulations, thereby validating the proposed approach.

Figure 3.6 Robot searching the book.

Figure 3.7 Robot performing the pushing trajectory.

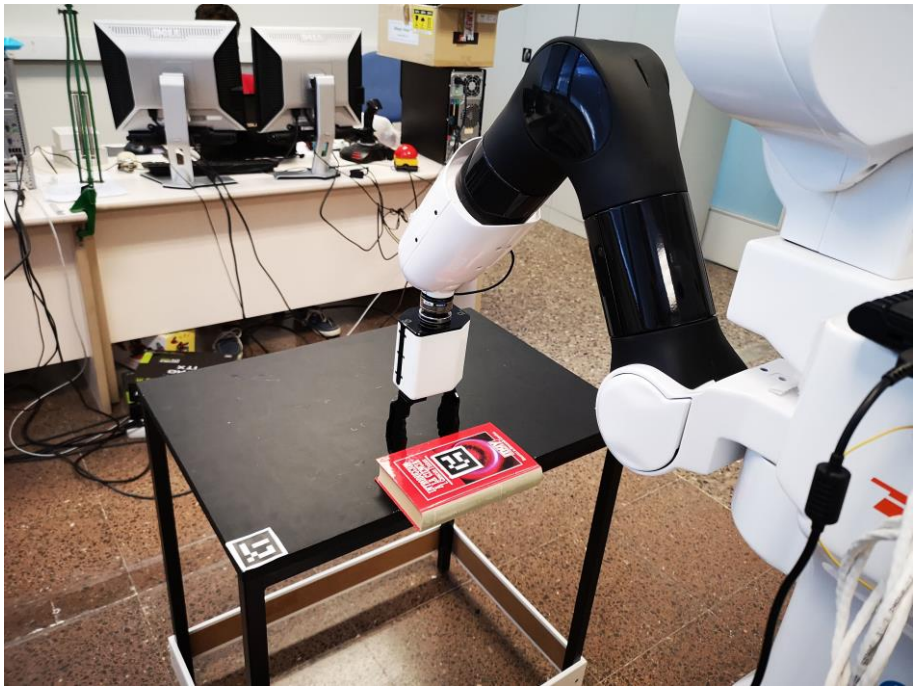

Figure 3.8 Robot at the end of the pushing trajectory.

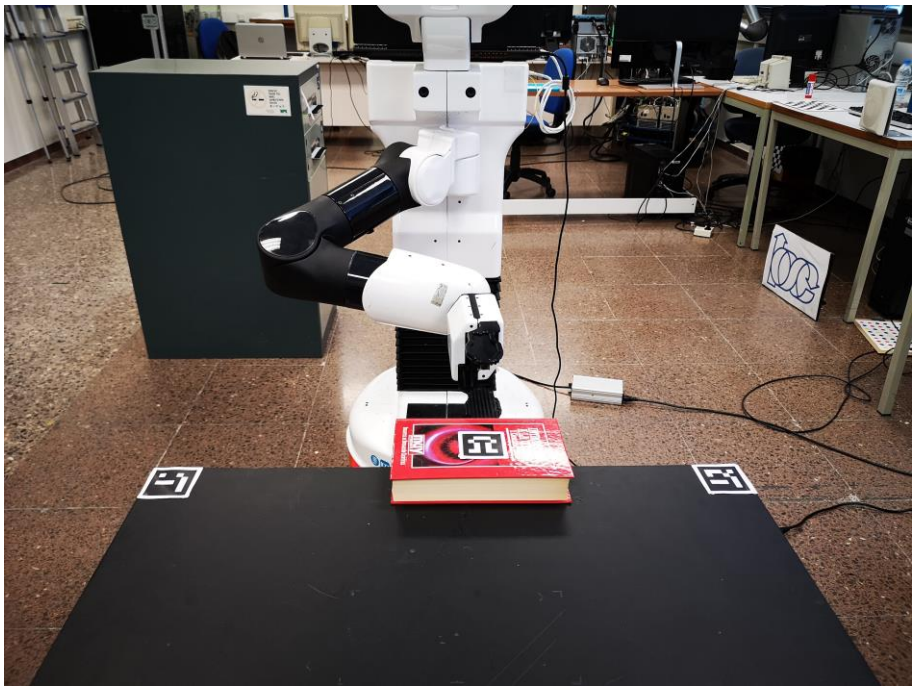

Figure 3.9 Robot in pre_grasping position.

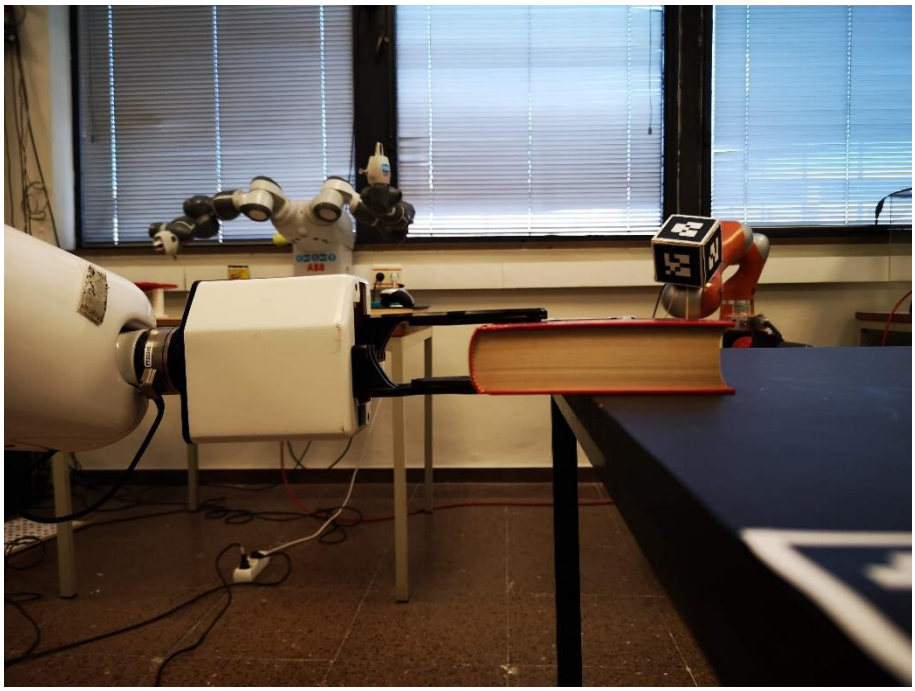

Figure 3.10 Book grasped.

Figure 3.11 Robot in safe_navigation position.

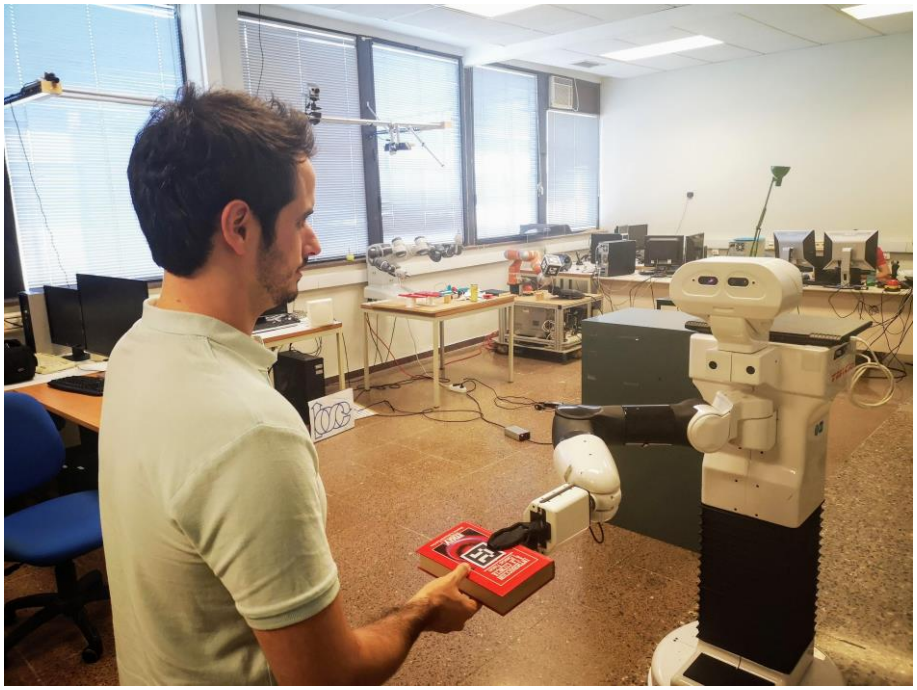

Figure 3.12 Robot offering the book to the user.

4. Costs

The total economic cost of the project is 4.666,75 €. Its computation, detailed below, includes the material, supplies and human resources.

The material used has been provided by the IOC robotics laboratory. The cost attributed to this project is due to the depreciation of this material. The material used was a TIAGo robot valued at 50,000€ and a computer of 2,000€. The amortization can be calculated from the total price of the robot and the computer, its lifespan and the total time of use. A robot with these characteristics has an estimated lifespan of 8 years. Considering a TIAGo usage of 6 hours a day, 5 days a week, 48 weeks a year, this generates a robot lifetime of 11520 hours. The laboratory computer used to develop the project has a useful life of 5 years. If the average time of use is 10 hours a day, its estimated useful life is 1,200 hours.

Taking into consideration that 362 hours have been invested in the project, the hours of use of each material are calculated as follows. The computer has been in use 90% of the time, being then its time of use of 325.8 hours. The robot has been in use 11% of the total time so its usage time is 40 hours. Taking into account the data presented on the cost of acquisition of the material, lifespan and time of use, the total cost of this item is 716.61€. The ETSEIB establishes for the students a recommended salary of 8 €/hour reason why taking into account the total of hours of the project the cost of this item supposes 2896 €. In addition, the time of IOC staff employed in supervising or guiding the project must be taken into account. It has been calculated that the time spent in this supervision has been 30 hours and taking an average cost of 35 € / h the cost of this item amounts to 1050 €.

Finally, the direct cost of electricity is calculated as the energy consumed by the computer and the robot. Taking into account the hours of operation of these equipment and their power consumption, the estimated consumption for the computer is 21.18kWh and for the robot is 6.4kWh. Considering that the average price of electricity in 2019 is 0.15 €/kWh, the total amount of electricity is 4.14 €.

5. Environmental and social impact

Any engineering project has an impact. The impact of the proposed project at the environmental and social level is presented below.

Since the aim of the project is to make the automation of libraries a generalized fact, it is considered that the environmental impact derived from the development of the solution is negligible in comparison with the impact implied by its implementation.

Considering that libraries tend to be supplied with clean energy since most of them depend on public institutions, the environmental impact of electricity consumption is minimal. For this reason, the main focus of environmental impact of this project is the manufacture and end of life of robots to be installed in libraries. In this sense, the use of recycled materials is recommended for the construction of the robots since the main impact comes from the extraction of the materials. On the other hand, it also recommends institutions that implement library automation to carry out a recycling plan for the materials of the robot once it ceases to be functional.

On a social level, this project allows an optimization of public resources as it contributes to the total automation of libraries. If the tasks with less added value that are developed in a library (such as the book gathering) are automated, the human resources can be used for tasks with more social return such as the promotion of reading and culture.

Conclusions

This project has proposed an approach that responds to the proposed objectives while considering the stated assumptions. The successful integration of several disciplines in the robotics field has allowed to achieve a complete solution. It is important to highlight that the proposed solution is robust to the perception and execution errors and allows to manage any orientation in which the book may lay on the table. Furthermore, through the refinement strategies applied, it has been possible to minimize the risks derived from the uncertainty in the detection of ArUco markers used for locating the table and the books.

As a result of this project, a robot is able to calculate how it should alter the orientation and position of a book in order to make the task of grasping possible with a parallel gripper which is a simple end-effector. Achieving the book grasping with a parallel gripper acquires a higher importance considering the project assumptions. As long as the maximum force that the robot can perform is not exceeded, a book of any size can be grasped without damaging the book. This is relevant because it discourages the use of direct grasping strategies that could be done with grippers shaped like a hand since the adaption to certain large books would be problematic. Also this type of gripper could provoke damages to the books such as folding its pages when performing the grasp. Another possibility would have been the use of pneumatic grippers but its use might not be appropriate for some book materials and could also reduce the social acceptance of the robot since the shape and the grasping principle of this gripper type differs significantly from the ones of the human hand.

Once the project has been completed, it can be seen that the resources used for its implementation were adequate. In particular, the TIAG robot used has allowed to achieve the objectives as it is a humanoid collaborative robot which may allow its easy integration in a real library and also has good features for object manipulation and for navigation. Moreover, by building the project on the Robot Operating System (ROS) framework, it has allowed, on the one hand, to implement all the required functionalities and, on the other hand, as it is an open-source platform and almost a standard in the robotics field, this project could be easily extended in the future.

Recapitulating, this project was conceived to contribute to the automation of book gathering in a library. It has effectively contributed to this purpose, although other tasks remain necessary for the full library automatization. Specifically discussing the completed task, future actions are suggested to extend it. It is recommended to work in the book and table recognition without the use of ArUco markers for a more natural integration in the libraries. It is also proposed to complement the project approach to be more robust with respect to the presence of obstacles in the gathering table.

Acknowledgements

Thanks to the Institute of Industrial and Control Engineering for giving me the opportunity to carry out this project and for all the received support. Special thanks to my project director, Dr. Raúl Suárez Feijóo and to Mr. Leopold Palomo Avellaneda for all their guidance and dedication.

I also want to deeply thank my sister, my mother, my father and my friends for all their support that has been fundamental to get me here. Finally, heartfelt thanks to my partner who, in addition to his unfailing moral support, has been a referent due to his expertise in robotics.

References

[1] OPEN SOURCE ROBOTICS FOUNDATION, *Robot Operating System* [http://ros.org, March 2019]

[2] ROMERO RAMIREZ, J. F., MUÑOZ SALINAS, R. and MEDINA CARNICER R. *Speeded up detection of squared fiducial markers, Image and Vision Computing.* Vol 76, 2010, p. 38-47.

[3] GARRIDO JURADO, R., ROMERO RAMIREZ, J. F., MUÑOZ SALINAS, R., MADRID CUEVAS, F. J. and MEDINA CARNICER R. *Generation of fiducial marker dictionaries using mixed integer linear programming, Pattern recognition.* Vol 51, 2016, p. 481-491.

[4] SICILIANO, B., SCIAVICCO L., VILLANI, L. and ORIOLO G. *Robotics: Modelling, Planning and Control.* Springer-Verlag London Limited 2009, chapter 2.

[5] OPEN SOURCE ROBOTICS FOUNDATION, MoveIt [https://moveit.ros.org, May 2019]

[6] KUFFNER, J. J. and LAVALLE S. M., *RRT-connect: An efficient approach to single-query path planning. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol.2, 2000, pp. 995-1001.

[7] DE MAEYER, J., MOYAERS, B. and DEMEESTER, E. *Cartesian path planning for arc welding robots: Evaluation of the Descartes algorithm Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* 2017, p. 1-8.

[8] PAL ROBOTICS S.L., *Tiago handbook.* Barcelona: 2018.

[9] OPEN SOURCE ROBOTICS FOUNDATION, *Gazebo simulator* [http://gazebosim.org, June 2019]